

ZTP for Factory Workflow

Overview	1
Prerequisites	3
Base	3
Networking	3
Storage	3
General	4
The Spokes YAML file	4
The workflow	8
OpenShift Pipelines installation	8
ZTPFW Pipelines	10
The Hub Pipeline	11
The Edge Node Pipeline	14
Post-Installation Configuration	17
Troubleshooting	18
Troubleshooting a PipelineRun	18
Debugging a task execution from the container in the cluster	19
Development	21
Deploying the environment in Virtual	21
Build Images	22
Executing a Pipeline Step	23

Overview

IMPORTANT

The ZTP for Factory Workflow images and code described in this document are for **Developer Preview** purposes and are **not supported** by Red Hat at this time.

ZTP for Factory Workflow provides a way for installing on top of OpenShift Container Platform the required pieces that will enable it to be used as a disconnected Hub Cluster and able to deploy Spoke Clusters that will be configured as the last step of the installation as disconnected too.

Using Openshift Container Agnostic Topology Integrated Chassis (OCATOPIC) to deploy ZTP for Factory Workflow, the whole procedure is managed and controlled end-to-end.

In this document, we'll cover the procedure and the automation around the different products and components that are involved in the ZTP for Factory workflow and orchestrated by OCATOPIC for a successful load deployment.

This repository contains the scripts and OpenShift Pipelines definitions used to configure a provided OpenShift cluster (reachable via `KUBECONFIG`) for use with the ZTP for Factory.

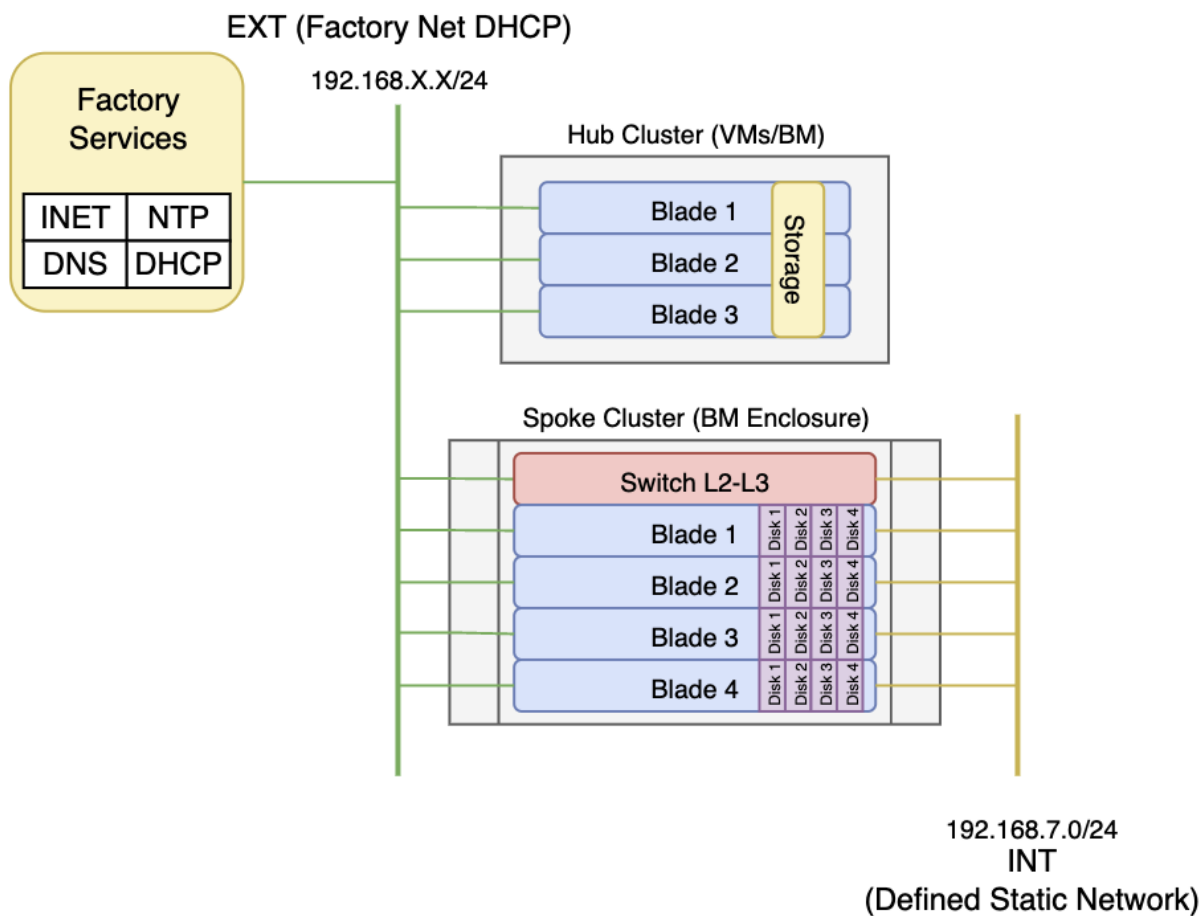
The pipeline will then cover several aspects:

- Create required components (ACM, Registry, etc...)
- Deploy and mirror a Registry with all required images and operators
- Configure ACM to provision the spokes (based on the `spokes.yaml` file) cluster and deploy all required components on it.
- Deploy Advanced Cluster Management (ACM) components
- etc...

The pipeline has two parts:

- One that deploys the HUB cluster configuration (based on existing requirements, like OCP deployed with ODF and volumes created)
- Another that deploys Spoke clusters based on the configuration `spokes.yaml` file using a HUB cluster configured with the previous pipeline.

The actual workflow and its details can be checked at the files inside the `pipelines` folder.



Prerequisites

Installer-provisioned installation of OpenShift Container Platform requires:

Base

- OpenShift Cluster with 3 masters
 1. All Cluster Operators in good health status
 2. Cluster reachable via a `KUBECONFIG` file
 3. The API/API-INT/Ingress should be deployed on the DHCP Ext Network (Factory network)

Networking

- DNS entries configured and resolvable from both internal and external network, with DNS on the DHCP Factory network
- HUB
 1. `api.<hub-domain>.<domain>` and `api-int.<hub-domain>.<domain>` entries to the same IP address
 2. ingress (`*.apps.<hub-domain>.<net-domain>`)
- SPOKE
 1. `api.<spoke-domain>.<net-domain>` and `api-int.<spoke-domain>.<net-domain>` entries to the same IP address
 2. ingress (`*.apps.<spoke-domain>.<net-domain>`)
- External DHCP with some free IPs on the factory to provide access to the Spoke using the external network interface
- Every Spoke will need at least ~6 IPs from this External Network (without the broadcast and network ip)
 1. 1 per node
 2. 1 API and same for API-INT
 3. 1 for the Ingress entry (`*.apps.<spoke-domain>.<net-domain>`)

Storage

- We need some Existant PVs on the HUB
 1. We cannot use `emptyDir` directive because between each step in the pipeline, the contents will be removed and we require them to further progress.
 2. 3 PVs for ACM (the expected size will depend on how many spokes will you deploy)
 3. 1 for the Hub Internal Registry, the base installation (which includes ACM, MetalLB, OCP version 4.X, NMState and some more images) we will need at least 200Gb on the Hub side (Maybe more if you have ODF/OCS deployed).

4. 1 for the HTTPD server, which will host the RHCOS images.
5. We need to fill the Openshift Storage requirements for the Hub like (SSD/NVME).
6. LSO should be enough but we recommend to use a more reliable one like ODF or NFS in order to avoid issues with the PVs and node scheduling pods.

General

- `spokes.yaml` file with the configuration for the spokes (In this initial version you will need to bake this file by hand)
- The enclosure is suppose to be just one Spoke which contains 3 masters, 1 worker and 1 Switch L2-L3

Of course, the requirements for the installation of OpenShift Container Platform are also to be satisfied on the hardware involved in the installation.

The Spokes YAML file

The `spokes.yaml` file contains all the configuration information required about the setup.

There's an example in the repo at <https://raw.githubusercontent.com/rh-ecosystem-edge/ztp-pipeline-relocatable/main/examples/config.yaml>

As you can check, it has two major sections `config` and `spokes` that will be explained in the next section.

Just keep in mind that the spokes section, can contain several `spoke-name` entries, one per spoke cluster to be deployed by the workflow.

Spokes.yaml walkthrough

Check next table for a commented configuration file with links to the explanation to each relevant file section and configuration value.

```
config:
  clusterimageset: openshift-v4.9.0
  OC_OCP_VERSION: "4.9"
  OC_OCP_TAG: "4.9.0-x86_64"
  OC_RHCOS_RELEASE: "49.84.202110081407-0"
  OC_ACM_VERSION: "2.4"
  OC_OCS_VERSION: "4.8"

spokes:
- spoke1-name:
    master0:
      nic_ext_dhcp: eno4
      nic_int_static: eno5
      mac_ext_dhcp: "aa:ss:dd:ee:b0:10"
```

```

    mac_int_static: "aa:ss:dd:ee:b1:10"
    bmc_url: "<url bmc>"
    bmc_user: "user-bmc"
    bmc_pass: "user-pass"
    root_disk: sda
    storage_disk:
      - sdb
      - sdc
      - sde
      - sdd
  master1:
    nic_ext_dhcp: eno4
    nic_int_static: eno5
    mac_ext_dhcp: "aa:ss:dd:ee:b0:11"
    mac_int_static: "aa:ss:dd:ee:b1:11"
    bmc_url: "<url bmc>"
    bmc_user: "user-bmc"
    bmc_pass: "user-pass"
    root_disk: sda
    storage_disk:
      - sdb
      - sdc
      - sde
      - sdd
  master2:
    nic_ext_dhcp: eno4
    nic_int_static: eno5
    mac_ext_dhcp: "aa:ss:dd:ee:b0:12"
    mac_int_static: "aa:ss:dd:ee:b1:12"
    bmc_url: "<url bmc>"
    bmc_user: "user-bmc"
    bmc_pass: "user-pass"
    root_disk: sda
    storage_disk:
      - sdb
      - sdc
      - sde
      - sdd
  worker0:
    nic_ext_dhcp: eno4
    nic_int_static: eno5
    mac_ext_dhcp: "aa:ss:dd:ee:b0:19"
    mac_int_static: "aa:ss:dd:ee:b1:19"
    bmc_url: "<url bmc>"
    bmc_user: "user-bmc"
    bmc_pass: "user-pass"
- spoke2-name:
  master0:
    nic_ext_dhcp: eno4
    nic_int_static: eno5
    mac_ext_dhcp: "aa:ss:dd:ee:b0:20"

```

```

mac_int_static: "aa:ss:dd:ee:b1:20"
bmc_url: "<url bmc>"
bmc_user: "user-bmc"
bmc_pass: "user-pass"
storage_disk:
  - sdb
  - sdc
  - sde
  - sdd
master1:
  nic_ext_dhcp: eno4
  nic_int_static: eno5
  mac_ext_dhcp: "aa:ss:dd:ee:b0:21"
  mac_int_static: "aa:ss:dd:ee:b1:21"
  bmc_url: "<url bmc>"
  bmc_user: "user-bmc"
  bmc_pass: "user-pass"
  storage_disk:
    - sdb
    - sdc
    - sde
    - sdd
master2:
  nic_ext_dhcp: eno4
  nic_int_static: eno5
  mac_ext_dhcp: "aa:ss:dd:ee:b0:22"
  mac_int_static: "aa:ss:dd:ee:b1:22"
  bmc_url: "<url bmc>"
  bmc_user: "user-bmc"
  bmc_pass: "user-pass"
  storage_disk:
    - sdb
    - sdc
    - sde
    - sdd
worker0:
  nic_ext_dhcp: eno4
  nic_int_static: eno5
  mac_ext_dhcp: "aa:ss:dd:ee:b0:29"
  mac_int_static: "aa:ss:dd:ee:b1:29"
  bmc_url: "<url bmc>"
  bmc_user: "user-bmc"
  bmc_pass: "user-pass"

```

Table 1. Required parameters

Parameter/Section	Description
<code>config</code>	This section marks the cluster configuration values that will be used for installation or configuration in both Hub and Spokes.

Parameter/Section	Description
<code>clusterimageset</code>	This setting defines the Cluster Image Set used for the HUB and the Spokes
<code>OC_OCP_VERSION</code>	Defines the OpenShift version to be used for the installation.
<code>OC_OCP_TAG</code>	This setting defines version tag to use
<code>OC_RHCOS_RELEASE</code>	This is the release to be used
<code>OC_ACM_VERSION</code>	Specifies which ACM version should be used for the deployment
<code>OC_OCS_VERSION</code>	This defines the OCS version to be used
<code>spokes</code>	This section is the one containing the configuration for each one of the Spoke Clusters
<code>spokename</code>	This option is configurable and will be the name to be used for the spoke cluster
<code>mastername</code>	This value must match <code>master0</code> , <code>master1</code> or <code>master2</code> .
<code>nic_ext_dhcp</code>	NIC connected to the external DHCP
<code>nic_int_static</code>	NIC interface name connected to the internal network
<code>mac_ext_dhcp</code>	MAC Address for the NIC connected to the external DHCP network
<code>mac_int_static</code>	MAC Address for the NIC connected to the internal static network
<code>bmc_url</code>	URL for the Baseboard Management Controller
<code>bmc_user</code>	Username for the BMC
<code>bmc_pass</code>	Password for the BMC
<code>root_disk</code>	Disk device to be used for OS installation
<code>storage_disk</code>	List of disk available in the node to be used for storage
<code>workername</code>	Hardcoded name as <code>worker0</code> for the worker node

The workflow

OpenShift Pipelines installation

First, we need to install OpenShift Pipelines Operator that will be used for running the pipeline, this is achieved by using a bootstrapping script that will install the Operator and the CR to initiate the deployment.

This script, will also create the required pipeline definitions and tasks.

Bootstrapping OpenShift Pipelines and ZTPFW

- Execute the bootstrap script file `pipelines/bootstrap.sh ${KUBECONFIG}` you can do that using this command:

NOTE

This bootstrap script will require at least these binaries: oc, yq, tkn

```
export KUBECONFIG=/root/.kcli/clusters/test-ci/auth/kubeconfig
curl -sLk https://raw.githubusercontent.com/rh-ecosystem-edge/ztp-pipeline-relocatable/main/pipelines/bootstrap.sh | bash -s
```

- An output similar to this one, will be shown:

[illegible]

This script will deploy Openshift-Pipelines and enabled the Tasks and Pipelines into the Hub cluster and more concretelly into the **spoke-deployer** Namespace.

We can now continue the flow using the command line or the UI to interact with OpenShift Pipelines, so it is recommended to install the Tekton CLI `tkn` to interact with OpenShift Pipelines from this [link](#).

ZTPFW Pipelines

We have 2 Pipelines created composed by some tasks each ones. Every Pipeline will be properly documented in each section.

Let's explain the pipeline arguments and Flags we use in the ZTPFW, for that we will use this sample command as a base:

```
tkn pipeline start -n spoke-deployer -p git-revision=main -p spokes-config="$(cat /root/amorgant/ztp-pipeline-relocatable/hack/deploy-hub-local/spokes.yaml)" -p kubeconfig=${KUBECONFIG} -w name=ztp,claimName=ztp-pvc --timeout 5h --use-param -defaults deploy-ztp-hub
```

Table 2. Pipeline Flags

Flag/Section	Description
<code>-n</code>	Openshift Namespace where the resources will be located
<code>-p</code>	Pipeline Parameter
<code>--timeout</code>	Pipeline General timeout.
<code>--use-param-defaults</code>	This means, "Apart of the parameters provided, the rest ones use the default options"
<code>-w</code>	The Workspace the place where Openshift Pipelines will hold the files during every step. We should not use an EmptyDir , if we use this option, the files generated between steps will not be saved. The best choice is <code>name=ztp,claimName=ztp-pvc</code> , this PVC will be created during the <code>bootstrap.sh</code> execution (It does not need more than 5Gb)

Table 3. Pipeline Arguments

Parameter/Section	Description	Required
Namespace	This is a hardcoded Namespace where all the Tasks and Pipelines will be deployed.	Yes
<code>git-revision</code>	This will download the ZTPFW code from a concrete branch. This is more for testing purposes. Default: <code>main</code>	No

Parameter/Section	Description	Required
<code>spokes-config</code>	This <code>spokes.yaml</code> file will contain the whole information about you wanna deploy at the same time. You need to put it with a <code>cat</code> command as we do in the example execution.	Yes
<code>kubeconfig</code>	This is the Hub kubeconfig that will be used during the pipeline execution. You can point to the file or just use the KUBECONFIG variable.	Yes
<code>-w name=ztp,claimName=ztp-pvc</code>	This is mandatory to be executed as we declare here. With this declaration we are telling Tekton to use the Workspace <code>ztp</code> and the content should be placed on the <code>ztp-pvc</code> Persistent Volume.	Yes
<code>Pipeline Name</code>	In our sample command, it's the last argument. We will tell Tekton to execute this Pipeline and it's equivalent to the Name. You can look at the Pipelines deployed using <code>tkn pr ls</code> or <code>tkn task ls</code> for the tasks	Yes

This command will trigger the Pipeline, it's asynchronous, and will respond with another command which access the log directly.

The Hub Pipeline

The Hub pipeline is an Openshift Object that will be used to deploy the infrastructure for the HUB to be ready to deploy spokes, it will need at least as we already explained before:

- An Openshift Hub Cluster available with 4.9 version
- The proper DNS entries and the API/Ingress exposed in the factory Network (DHCP Ext Network)
- At least 6 PVs availables ready to be bound.
- Internet connectivity
- Openshift Pipelines already there (explained in the past section)

Ok now we can continue with the Pipeline execution, it will be a command like this:

```
tkn pipeline start -n spoke-deployer -p git-revision=main -p spokes-config="$(cat /root/amorgant/ztp-pipeline-relocatable/hack/deploy-hub-local/spokes.yaml)" -p kubeconfig=${KUBECONFIG} -w name=ztp,claimName=ztp-pvc --timeout 5h --use-param -defaults deploy-ztp-hub
```

After this command you will see this on the prompt:

```
PipelineRun started: deploy-ztp-hub-run-w5k7l
```

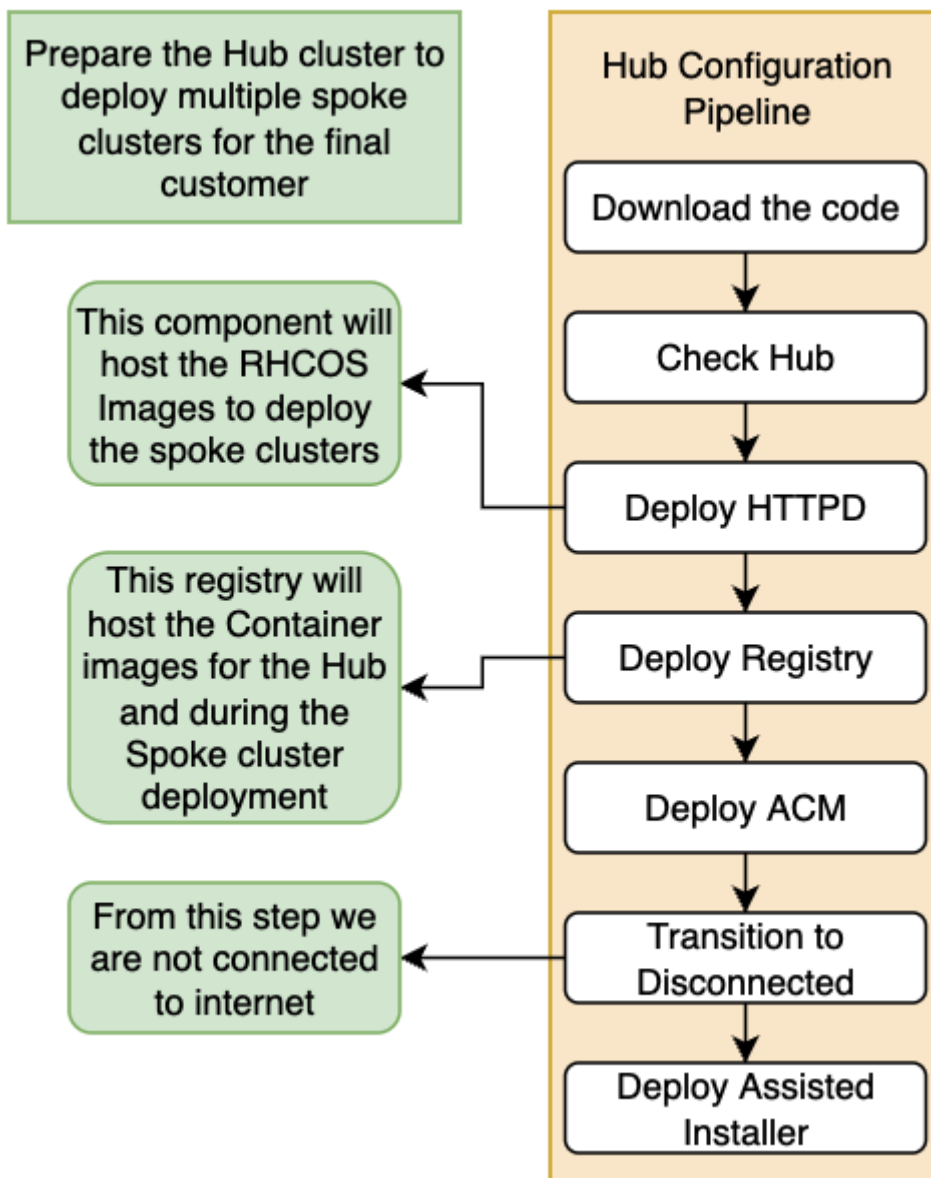
In order to track the PipelineRun progress run:

```
tkn pipelinerun logs deploy-ztp-hub-run-w5k7l -f -n spoke-deployer
```

If you check the logs of just follow the PipelineRun on the Openshift Console, you will see every step the Pipeline will follow. Let's explain which steps we have and what they do:

The Workflow

The Hub Workflow will be something like what you're seeing in the image, let's dig a bit on every step



- **Download the code:** This phase will be mandatory or not, depending the scenario, if the environemnt it's fully disconnected, this source code will be embedded into the Container Image.
- **Check the Hub cluster:** We will ensure all the things are ready to start the Hub provisioning, things like ClusterOperators, ClusterVersion and Nodes up and ready are basic to start working.
- **Deploy HTTPD Server:** In this step we will deploy an HTTPD server in order to host the RHCOS Images that will be used in the Spoke deployment as a base to bake the customized ISOS for the Bare Metal nodes.
- **Deploy Image Registry:** We will host an internal registry in the Hub cluster, after that perform the Sync between the OCP and OLM images from Quay/RedHat registries to the internal one. This step will enable, in a future step, to change the ICSP and Catalog sources that allows us to use the this Hub cluster as an isolated one. Last step here is update the OCP global PullSecret to give the cluster the capability to access the images.
- **Deploy ACM:** We will deploy the ACM into this Hub cluster which is the piece that allow us to deploy the Edge clusters
- **Transition to Disconnected:** Here we will deploy the ICSP and the Catalog sources for the hub

to point to himself as a source of the images and operators.

- **Deploy Assisted Installer:** This is part of ACM which is not deployed by default. Here we configure the way the Edge cluster will be deployed, certificates, image sources, cluster details, etc...

The Edge Node Pipeline

The Edge Node Pipeline is an Openshift Object that will be used to deploy the Spoke clusters (Spokes only on the factory, after that they will be typical Edge nodes).

We will need some prerequisites here:

- Enough DHCP IPs in the external/factory network to hold the Edge Cluster
- The API, API-INT and Ingress entries
 1. api.<spoke-cluster-name>.<network-domain>
 2. api-int.<spoke-cluster-name>.<network-domain>
 3. *.apps.<spoke-cluster-name>.<network-domain>
- Clean disks for the OCS/ODF StorageCluster
 1. If the disks are not clean we will provide the way to clean them in other section.
- DNS Resolution between the Spoke and the Hub API/Ingress entries.

This is how we execute the Pipeline:

```
tkn pipeline start -n spoke-deployer -p git-revision=tekton -p spokes-config="$(cat /root/jparrill/ztp-pipeline-relocatable/hack/deploy-hub-local/spokes.yaml)" -p kubeconfig=${KUBECONFIG} -w name=ztp,claimName=ztp-pvc --timeout 5h --use-param -defaults deploy-ztp-spokes
```

After this command you will see this on the prompt:

```
PipelineRun started: deploy-ztp-spokes-run-w5k7l
```

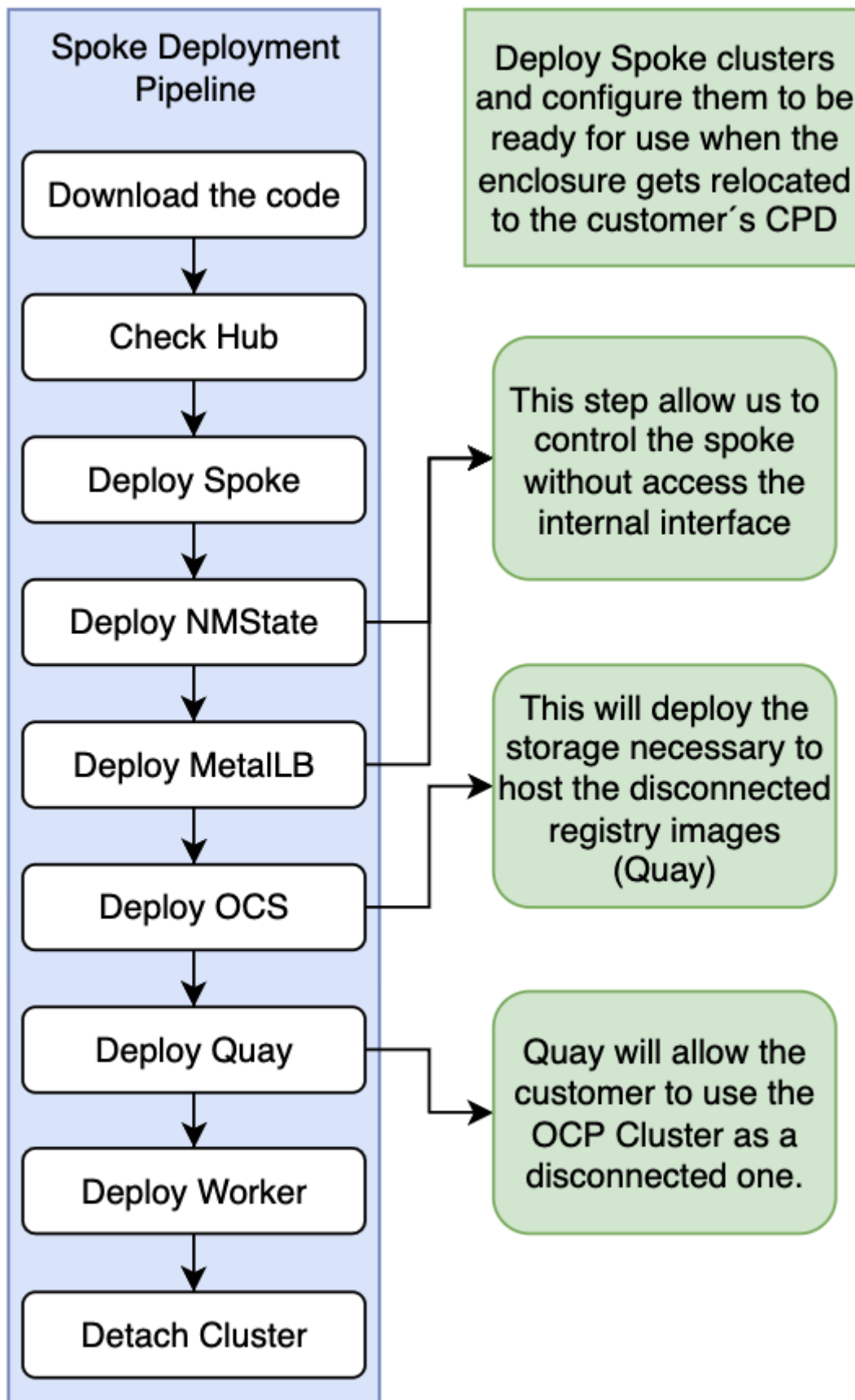
In order to track the PipelineRun progress run:

```
tkn pipelinerun logs deploy-ztp-spokes-run-w5k7l -f -n spoke-deployer
```

If you check the logs of just follow the PipelineRun on the Openshift Console, you will see every step the Pipeline will follow. Let's explain which steps we have and what they do:

The Workflow

The Hub Workflow will be something like what you're seeing in the image, let's dig a bit on every step



- **Download the code:** This phase will be mandatory or not, depending the scenario, if the environemnt it's fully disconnected, this source code will be embbeded into the Container Image.
- **Check the Hub cluster:** We will ensure all the things are ready to start the Hub provisioning, things like ClusterOperators, ClusterVersion and Nodes up and ready are basic to start working.

- **Deploy Spoke Cluster:** The Pipeline will start with the Edge cluster Provisioning. This process will end pushing a notification from the Edge cluster to the Hub and answering with an ACK.
- ***Deploy NMState and MetalLB:** This is one of the Key steps, without this you will not be able to access the API/Ingress using the external address. This step deploys NMState and MetalLB operators which creates 1 profile per node to grab an IP from external's Network DHCP, then the MetalLB will create a resource called AddressPool to perform the relationship between the internal and external interface using a LoadBalancer interface. And finally creating a Service for the API and the Ingress.
- **Deploy OCS/ODF:** This step will deploy Local Storage Operator and also Openshift Storage. Local Storage Operator will use the node disks (NVMEs) to create PVs, which OCS will use to deploy the StorageCluster on top of them, to generate the Storage Classes and Dynamic provisioning of the PVs.
- **Deploy Quay:** We will deploy Quay Operator and components of Quay because the final customer will need a fully supported solution in the Edge and the factory (in the most probable scenario) will have their own internal registry in the factory. This Quay deployment has a small footprint enabling only the things needed to host an Internal Registry with basic functions.
- **Deploy Worker Node:** At this point we will deploy the Worker node, and we will make it join to the Edge cluster.
- **Detach Edge Cluster:** This final step will perform some actions, first ensure that the things are well set and working. After that it will save the SSH-RSA keys, Kubeconfig and Kubeadmin password into the Hub, more concretely in the <spoke-cluster-name> Namespace in the hub cluster. This could be sent afterwards to the customer, this policy should be set by the factory.

Post-Installation Configuration

After successfully deploying an installer-provisioned cluster, consider the following post-installation procedures.

Troubleshooting

Troubleshooting a PipelineRun

To debug the Hub Pipeline you just need to

- Grab PipelineRuns executed

```
export KUBECONFIG=<PATH TO KUBECONFIG>
tkn pr ls
```

NAME	STARTED	DURATION	STATUS
deploy-ztp-spokes-run-wll7j	3 days ago	2 hours	Succeeded
deploy-ztp-spokes-run-f9k4l	3 days ago	5 minutes	Cancelled(PipelineRunCancelled)
deploy-ztp-spokes-run-f4g2h	3 days ago	3 minutes	Cancelled(PipelineRunCancelled)
deploy-ztp-hub-run-6n4vr	3 days ago	2 minutes	Succeeded
deploy-ztp-spokes-run-2kr75	3 days ago	2 hours	Succeeded
deploy-ztp-spokes-run-8wkh4	3 days ago	13 minutes	Cancelled(PipelineRunCancelled)
deploy-ztp-spokes-run-5b9xp	4 days ago	2 hours	Succeeded
deploy-ztp-spokes-run-vjwpd	4 days ago	18 minutes	Cancelled(PipelineRunCancelled)
deploy-ztp-hub-run-qp6vf	4 days ago	29 minutes	Succeeded
deploy-ztp-hub-run-s5tjs	4 days ago	3 minutes	Failed

- Grab the ID and find the task which failed

```
tkn pr describe deploy-ztp-spokes-run-wll7j
```

Taskruns				
NAME	TASK NAME	STARTED	DURATION	STATUS
• deploy-ztp-spokes-run-wll7j-detach-cluster-ptswr	detach-cluster	2 days ago	19 seconds	Succeeded
• deploy-ztp-spokes-run-wll7j-deploy-ui-2gnm	deploy-ui	2 days ago	35 seconds	Succeeded
• deploy-ztp-spokes-run-wll7j-deploy-workers-9rbf9	deploy-workers	2 days ago	6 minutes	Succeeded
• deploy-ztp-spokes-run-wll7j-deploy-icsp-spokes-post-jbnc2	deploy-icsp-spokes-post	2 days ago	8 minutes	Succeeded
• deploy-ztp-spokes-run-wll7j-deploy-disconnected-registry-8x8p7	deploy-disconnected-registry	3 days ago	1 hour	Succeeded
• deploy-ztp-spokes-run-wll7j-deploy-ocs-pbvpb	deploy-ocs	3 days ago	5 minutes	Succeeded
• deploy-ztp-spokes-run-wll7j-deploy-metallb-hstkl	deploy-metallb	3 days ago	2 minutes	Succeeded
• deploy-ztp-spokes-run-wll7j-deploy-icsp-spokes-pre-jvsbv	deploy-icsp-spokes-pre	3 days ago	42 seconds	Succeeded
• deploy-ztp-spokes-run-wll7j-deploy-spokes-pm7qv	deploy-spokes	3 days ago	32 minutes	Succeeded
• deploy-ztp-spokes-run-wll7j-pre-flight-cp9p5	pre-flight	3 days ago	8 seconds	Succeeded
• deploy-ztp-spokes-run-wll7j-fetch-from-git-ngfpf	fetch-from-git	3 days ago	9 seconds	Succeeded

- Grab the Task ID which failed and describe it

```
tkn tr describe deploy-ztp-spokes-run-wll7j-detach-cluster-ptswr
```

```

Name:      deploy-ztp-spokes-run-wll7j-detach-cluster-ptswr
Namespace: spoke-deployer
Task Ref:  spoke-detach-cluster
Service Account: pipeline
Timeout:   5h0m0s
Labels:
  app.kubernetes.io/managed-by=tekton-pipelines
  tekton.dev/memberOf=tasks
  tekton.dev/pipeline=deploy-ztp-spokes
  tekton.dev/pipelineRun=deploy-ztp-spokes-run-wll7j
  tekton.dev/pipelineTask=detach-cluster
  tekton.dev/task=spoke-detach-cluster

```

Status

STARTED	DURATION	STATUS
2 days ago	19 seconds	Succeeded

Input Resources

No input resources

Output Resources

No output resources

Params

NAME	VALUE
spokes-config	config:
clusterimageset	openshift-v4.9.13
OC_OCP_VERSION	'4.9'
OC_OCP_TAG	'4.9.13-x86_64'
OC_RHCOS_RELEASE	'49.84.202110081407-0' # TODO automate it to get it automated using binary
OC_ACM_VERSION	'2.4'
OC_OCS_VERSION	'4.8'

spokes:

```

- spoke0-cluster:

```

Debugging a task execution from the container in the cluster

```

[root@flaper87-baremetal02 ~]# oc get pod -n spoke-deployer
NAME READY STATUS RESTARTS AGE
deploy-ztp-hub-run-96tnl-deploy-disconnected-registry-4m2-5ts85 2/4 NotReady 0 6m32s
deploy-ztp-hub-run-96tnl-deploy-httpd-server-rlrwq-pod-wsh5k 0/1 Completed 0 6m41s
deploy-ztp-hub-run-96tnl-fetch-from-git-zl7m5-pod-fck69 0/1 Completed 0 6m59s
deploy-ztp-hub-run-96tnl-pre-flight-rgdtr-pod-2gmh6 0/1 Completed 0 6m50s

```

```
[root@flaper87-baremetal02 ~]# oc debug pod/deploy-ztp-hub-run-96tnl-deploy-  
disconnected-registry-4m2-5ts85 -n spoke-deployer  
Defaulting container name to step-deploy-disconnected-registry.  
Use 'oc describe pod/deploy-ztp-hub-run-96tnl-deploy-disconnected-registry-4m2-5ts85-  
debug -n spoke-deployer' to see all of the containers in this pod.
```

```
Starting pod/deploy-ztp-hub-run-96tnl-deploy-disconnected-registry-4m2-5ts85-debug,  
command was: /tekton/tools/entrypoint -wait_file /tekton/downward/ready  
-wait_file_content -post_file /tekton/tools/0 -termination_path /tekton/termination  
-step_metadata_dir /tekton/steps/step-deploy-disconnected-registry  
-step_metadata_dir_link /tekton/steps/0 -docker-cfg=pipeline-dockercfg-w6x1w  
-entrypoint /tekton/scripts/script-0-x6mfw --
```

```
Pod IP: 10.134.0.60
```

If you don't see a command prompt, try pressing enter.

```
sh-4.4# cd /workspace/ztp/
```

Development

NOTE

This documentation it's mostly for the developers/qes etc... working in the project.

Deploying the environment in Virtual

This is a very expensive option to work with all nodes in virtual, which means, you will need a big boy to make this work:

Hardware requirements

Hardware Reqs for the Hub (3 Nodes):

- CPUs: 48 (16 each)
- RAM: 54 Gbs (18 each)
- Storage: 300 Gbs (each)

Hardware Reqs for the Edge Cluster (3 Master + 1 Worker Nodes):

Master Nodes:

- CPUs: 72 (24 each)
- RAM: 192 (64 each)
- Storage: 4 extra disks with 200Gb each one

Worker Node: - CPUs: 12 - RAM: 16 - Storage: 4 extra disks with 200Gb each one

Software requirements

- Libvirt/Qemu/KVM
- Kcli for the scripts.
- Some binaries oc, kubectl, tkn, yq, jq and ketall (for debugging)

Deploying the Base Hub

Deploys the Hub cluster with an NFS as a Base Storage for the requirements

```
git clone git@github.com:rh-ecosystem-edge/ztp-pipeline-relocatable.git
cd ztp-pipeline-relocatable/hack/deploy-hub-local
./build-hub.sh ${HOME}/openshift_pull.json 1
```

Bootstrapping Openshift Pipelines

Installs the necessary things to start executing the Pipelines

```
export KUBECONFIG=/root/.kcli/clusters/test-ci/auth/kubeconfig
curl -sLk https://raw.githubusercontent.com/rh-ecosystem-edge/ztp-pipeline-relocatable/main/pipelines/bootstrap.sh | bash -s
```

Executing the Hub Pipeline

You can customize the parameter `git-revision=<BRANCH>` to point to your own branch

```
export KUBECONFIG=/root/.kcli/clusters/test-ci/auth/kubeconfig
tkn pipeline start -n spoke-deployer -p git-revision=main -p spokes-config="$(cat /root/amorgant/ztp-pipeline-relocatable/hack/deploy-hub-local/spokes.yaml)" -p kubeconfig=${KUBECONFIG} -w name=ztp,claimName=ztp-pvc --timeout 5h --use-param -defaults deploy-ztp-hub
```

Creating the Edge Cluster VMs

Creates 4 VMs and the proper DNS entries for the involved network

```
./build-spoke.sh ${HOME}/openshift_pull.json 1
```

Executing the Edge Cluster Pipeline

You can customize the parameter `git-revision=<BRANCH>` to point to your own branch

```
export KUBECONFIG=/root/.kcli/clusters/test-ci/auth/kubeconfig
tkn pipeline start -n spoke-deployer -p git-revision=main -p spokes-config="$(cat /root/amorgant/ztp-pipeline-relocatable/hack/deploy-hub-local/spokes.yaml)" -p kubeconfig=${KUBECONFIG} -w name=ztp,claimName=ztp-pvc --timeout 5h --use-param -defaults deploy-ztp-spokes
```

Build Images

You will need first access to the Quay Organization called **ZTPFW**, just ask whoever people involved in the project.

You have some targets already in the Makefile, and today you just need to execute:

```
make
```

this will change in the future to add functionality to the Image building

Executing a Pipeline Step

Imagine you have an environment already deployed and you need to test the step you are working on, think on for example the UI. For that you just need to:

- First step you updates the code in the PVC (This can change in the future when we embed the code in the Container Image)

```
tkn task start -n spoke-deployer -p git-revision=<YOUR BRANCH> -p spokes-config="$(cat /root/jparrill/ztp-pipeline-relocatable/hack/deploy-hub-local/spokes.yaml)" -p kubeconfig=${KUBECONFIG} -w name=ztp,claimName=ztp-pvc --timeout 5h --use-param -defaults fetch-from-git
```

- This second one executes the Pipeline Step

```
tkn task start -n spoke-deployer -p git-revision=<YOUR BRANCH> -p spokes-config="$(cat /root/jparrill/ztp-pipeline-relocatable/hack/deploy-hub-local/spokes.yaml)" -p kubeconfig=${KUBECONFIG} -w name=ztp,claimName=ztp-pvc --timeout 5h --use-param -defaults spoke-deploy-ui
```